

Deletion Operations on Deterministic Families of Automata<sup>☆,☆☆</sup>Joey Eremondi<sup>a,2</sup>, Oscar H. Ibarra<sup>b,1</sup>, Ian McQuillan<sup>c,2,\*</sup><sup>a</sup>*Department of Information and Computing Sciences  
Utrecht University, P.O. Box 80.089 3508 TB Utrecht, The Netherlands*<sup>b</sup>*Department of Computer Science**University of California, Santa Barbara, CA 93106, USA*<sup>c</sup>*Department of Computer Science, University of Saskatchewan  
Saskatoon, SK S7N 5A9, Canada*

---

**Abstract**

Many different deletion operations are investigated applied to languages accepted by one-way and two-way deterministic reversal-bounded multicounter machines, deterministic pushdown automata, and finite automata. Operations studied include the prefix, suffix, infix and outfix operations, as well as left and right quotient with languages from different families. It is often expected that language families defined from deterministic machines will not be closed under deletion operations. However, here, it is shown that one-way deterministic reversal-bounded multicounter languages are closed under right quotient with languages from many different language families; even those defined by nondeterministic machines such as the context-free languages. Also, it is shown that when starting with one-way deterministic machines with one counter that makes only one reversal, taking the left quotient with languages from many different language families — again including those defined by nondeterministic machines such as the context-free languages — yields only one-way deterministic reversal-bounded multicounter languages (by increasing the number of counters). These results are surprising given the nondeterministic nature of the deletion operation. However, if there are two more reversals on the counter, or a second 1-reversal-bounded counter, taking the left quotient (or even just the suffix operation) yields languages that can neither be accepted by deterministic reversal-bounded multicounter machines, nor by 2-way nondeterministic machines with one reversal-bounded counter.

**Keywords:** Automata and Logic, Counter Machines, Deletion Operations, Reversal-Bounds, Determinism, Finite Automata

---

**1. Introduction**

This paper involves the study of various types of deletion operations applied to languages accepted by deterministic classes of machines. Deletion operations, such as left and right quotients, and word operations such as prefix, suffix, infix, and outfix, are more commonly studied applied to languages accepted by classes of nondeterministic machines. Indeed, many language families accepted by nondeterministic acceptors form *full trios* (closure under homomorphism, inverse homomorphism, and intersection with regular languages), and every full trio is closed under left and right quotient with regular languages, prefix, suffix, infix, and outfix [2]. For families of languages accepted by deterministic machines however, the situation is more

---

<sup>☆</sup>This is an extended version of the conference paper [1].

<sup>☆☆</sup>©2016. This manuscript version is made available under the CC-BY-NC-ND 4.0 license <http://creativecommons.org/licenses/by-nc-nd/4.0/>

\*Corresponding author

URL: [j.s.eremondi@students.uu.nl](mailto:j.s.eremondi@students.uu.nl) (Joey Eremondi), [ibarra@cs.ucsb.edu](mailto:ibarra@cs.ucsb.edu) (Oscar H. Ibarra), [mcquillan@cs.usask.ca](mailto:mcquillan@cs.usask.ca) (Ian McQuillan)

<sup>1</sup>Supported, in part, by NSF Grant CCF-1117708 (Oscar H. Ibarra).

<sup>2</sup>Supported, in part, by Natural Sciences and Engineering Research Council of Canada Grant 327486-2010 (Ian McQuillan).

tricky due to the nondeterministic behaviour of the deletion. Indeed, deterministic pushdown automata are not even closed under left quotient with a set of individual letters. Here, most deterministic machine models studied will involve restrictions of one-way deterministic reversal-bounded multicounter machines (DCM). These are machines that operate like deterministic finite automata with an additional fixed number of counters, where there is a bound on the number of times each counter switches between increasing and decreasing [3, 4]. The family  $\text{DCM}(k, l)$  consists of languages accepted by machines with  $k$  counters that are  $l$ -reversal-bounded. DCM languages have many decidable properties, such as emptiness, infiniteness, equivalence, inclusion, universe, and disjointness [4]. Furthermore,  $\text{DCM}(1, l)$  forms an important restriction of deterministic pushdown automata.

These machines have been studied in a variety of different applications, such as to membrane computing [5], verification of infinite-state systems [6, 7, 8, 9], and Diophantine equations [9].

Recently, in [10], a related study was conducted for insertion operations; specifically operations defined by ideals obtained from the prefix, suffix, infix, and outfix relations, as well as left and right concatenation with languages from different language families. It was found that languages accepted by one-way deterministic reversal-bounded counter machines with one reversal-bounded counter are closed under right concatenation with  $\Sigma^*$ , but having two 1-reversal-bounded counters and right concatenating  $\Sigma^*$  yields languages outside of both DCM and  $2\text{DCM}(1)$  (languages accepted by two-way deterministic machines with one counter that is reversal-bounded). It also follows from this analysis that the right input end-marker is necessary for even one-way deterministic reversal-bounded counter machines, when there are at least two counters. Furthermore, concatenating  $\Sigma^*$  to the left of some one-way deterministic 1-reversal-bounded one counter languages yields languages that are neither in DCM nor  $2\text{DCM}(1)$ . Other recent results on reversal-bounded multicounter languages include a technique to show languages are outside of DCM [11]. Closure properties of nondeterministic counter machines under other types of deletion operations were studied in [12].

In this paper we investigate closure properties of types of deterministic machines. In Section 2, preliminary background and notation are introduced. In Section 3, erasing operations where DCM is closed are studied. It is shown that DCM is closed under right quotient with context-free languages, and that the left quotient of  $\text{DCM}(1, 1)$  by a context-free language is in DCM. Both results are generalizable to quotients with a variety of different families of languages containing only semilinear languages. In Section 4, non-closure of DCM under erasing operations are studied. It is shown that the set of suffixes, infixes, or outfixes of a  $\text{DCM}(1, 3)$  or  $\text{DCM}(2, 1)$  language can be outside of both DCM and  $2\text{DCM}(1)$ . In Section 5, DPCMs (deterministic pushdown automata augmented by reversal-bounded counters), and NPCMs (the nondeterministic variant) are studied. It is shown that DPCM is not closed under prefix or suffix, and the right or left quotient of the language accepted by a 1-reversal-bounded deterministic pushdown automaton by a  $\text{DCM}(1, 1)$  language can be outside DPCM. In Section 6, the effective closure of regular languages with other families is briefly discussed, and in Section 7, bounded languages are discussed.

## 2. Preliminaries

The set of non-negative integers is denoted by  $\mathbb{N}_0$ , and the set of positive integers by  $\mathbb{N}$ . For  $c \in \mathbb{N}_0$ , let  $\pi(c)$  be 0 if  $c = 0$ , and 1 otherwise.

We assume knowledge of standard formal language theoretic concepts such as finite automata, determinism, nondeterminism, semilinearity, recursive, and recursively enumerable languages [3, 13]. Next, we will give some notation used in the paper. The empty word is denoted by  $\lambda$ . If  $\Sigma$  is a finite alphabet, then  $\Sigma^*$  is the set of all words over  $\Sigma$  and  $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$ . For a word  $w \in \Sigma^*$ , if  $w = a_1 \cdots a_n$  where  $a_i \in \Sigma$ ,  $1 \leq i \leq n$ , the length of  $w$  is denoted by  $|w| = n$ , and the reversal of  $w$  is denoted by  $w^R = a_n \cdots a_1$ , which is extended to reversals of languages in the natural way. In addition, if  $a \in \Sigma$ ,  $|w|_a$  is the number of  $a$ 's in  $w$ . A language over  $\Sigma$  is any subset of  $\Sigma^*$ . Given a language  $L \subseteq \Sigma^*$ , the complement of  $L$ ,  $\Sigma^* \setminus L$  is denoted by  $\bar{L}$ . Given two languages  $L_1, L_2$ , the left quotient of  $L_2$  by  $L_1$ ,  $L_1^{-1}L_2 = \{y \mid xy \in L_2, x \in L_1\}$ , and the right quotient of  $L_1$  by  $L_2$  is  $L_1L_2^{-1} = \{x \mid xy \in L_1, y \in L_2\}$ . A *full trio* is a language family closed under homomorphism, inverse homomorphism, and intersection with regular languages [13].

Let  $n \in \mathbb{N}$ . Then  $Q \subseteq \mathbb{N}_0^n$  is a *linear* set if there is a vector  $\vec{c} \in \mathbb{N}_0^n$  (the constant vector), and a set of vectors  $V = \{\vec{v}_1, \dots, \vec{v}_r\}$ ,  $r \geq 0$ , each  $\vec{v}_i \in \mathbb{N}_0^n$  such that  $Q = \{c + t_1\vec{v}_1 + \dots + t_r\vec{v}_r \mid t_1, \dots, t_r \in \mathbb{N}_0\}$ . A finite union of linear sets is called a *semilinear set*.

A language  $L$  is *word-bounded* or simply *bounded* if  $L \subseteq w_1^* \dots w_k^*$  for some  $k \geq 1$  and (not-necessarily distinct) words  $w_1, \dots, w_k$ . Further,  $L$  is *letter-bounded* if each  $w_i$  is a letter. Also,  $L$  is *bounded-semilinear* if  $L \subseteq w_1^* \dots w_k^*$  and  $Q = \{(i_1, \dots, i_k) \mid w_1^{i_1} \dots w_k^{i_k} \in L\}$  is a semilinear set [14].

We now present notation for common word and language operations used throughout the paper.

**Definition 1.** For a language  $L \subseteq \Sigma^*$ , the prefix, suffix, infix, and outfix operations are defined by:

- $\text{pref}(L) = \{w \mid wx \in L, x \in \Sigma^*\},$
- $\text{suff}(L) = \{w \mid xw \in L, x \in \Sigma^*\},$
- $\text{inf}(L) = \{w \mid xwy \in L, x, y \in \Sigma^*\},$
- $\text{outf}(L) = \{xy \mid xwy \in L, w \in \Sigma^*\}.$

Note that  $\text{pref}(L) = L(\Sigma^*)^{-1}$  and  $\text{suff}(L) = (\Sigma^*)^{-1}L$ .

The outfix operation has been generalized to the notion of embedding [15]:

**Definition 2.** The  $m$ -embedding of a language  $L \subseteq \Sigma^*$  is the following set:  $\text{emb}(L, m) = \{w_0 \dots w_m \mid w_0 x_1 \dots w_{m-1} x_m w_m \in L, w_i \in \Sigma^*, 0 \leq i \leq m, x_j \in \Sigma^*, 1 \leq j \leq m\}.$

Note that  $\text{outf}(L) = \text{emb}(L, 1)$ .

A *nondeterministic multicounter machine* is a finite automaton augmented by a fixed number of counters. The counters can be increased, decreased, tested for zero, or tested to see if the value is positive. A multicounter machine is *reversal-bounded* if there exists a fixed  $l$  such that, in every accepting computation, the count on each counter alternates between increasing and decreasing at most  $l$  times.

Formally, a *one-way  $k$ -counter machine* is a tuple  $M = (k, Q, \Sigma, \triangleleft, \delta, q_0, F)$ , where  $Q, \Sigma, \triangleleft, q_0, F$  are respectively the finite set of states, the input alphabet, the right input end-marker, the initial state in  $Q$ , and the set of final states that is a subset of  $Q$ . The transition function  $\delta$  (defined as in [4] except with only a right end-marker since we only use one-way inputs) is a relation from  $Q \times (\Sigma \cup \{\triangleleft\}) \times \{0, 1\}^k$  into  $Q \times \{S, R\} \times \{-1, 0, +1\}^k$ , such that if  $\delta(q, a, c_1, \dots, c_k)$  contains  $(p, d, d_1, \dots, d_k)$  and  $c_i = 0$  for some  $i$ , then  $d_i \geq 0$  to prevent negative values in any counter. The direction of the input tape head movement is given by the symbols  $S$  and  $R$  for either *stay* or *right* respectively. The machine  $M$  is *deterministic* if  $\delta$  is a partial function. A *configuration* of  $M$  is a  $k + 2$ -tuple  $(q, w\triangleleft, c_1, \dots, c_k)$  for describing the situation where  $M$  is in state  $q$ , with  $w \in \Sigma^*$  still to read as input, and  $c_1, \dots, c_k \in \mathbb{N}_0$  are the contents of the  $k$  counters. The derivation relation  $\vdash_M$  is defined between configurations, where  $(q, aw, c_1, \dots, c_k) \vdash_M (p, w', c_1 + d_1, \dots, c_k + d_k)$ , if  $(p, d, d_1, \dots, d_k) \in \delta(q, a, \pi(c_1), \dots, \pi(c_k))$  where  $d \in \{S, R\}$  and  $w' = aw$  if  $d = S$ , and  $w' = w$  if  $d = R$ . Extended derivations are given by  $\vdash_M^*$ , the reflexive, transitive closure of  $\vdash_M$ . A word  $w \in \Sigma^*$  is accepted by  $M$  if  $(q_0, w\triangleleft, 0, \dots, 0) \vdash_M^* (q, \triangleleft, c_1, \dots, c_k)$ , for some  $q \in F$ , and  $c_1, \dots, c_k \in \mathbb{N}_0$ . The language accepted by  $M$ , denoted by  $L(M)$ , is the set of all words accepted by  $M$ . The machine  $M$  is  $l$ -reversal bounded if, in every accepting computation, the count on each counter alternates between increasing and decreasing at most  $l$  times.

We denote by  $\text{NCM}(k, l)$  the family of languages accepted by one-way nondeterministic  $l$ -reversal-bounded  $k$ -counter machines. We denote by  $\text{DCM}(k, l)$  the family of languages accepted by one-way deterministic  $l$ -reversal-bounded  $k$ -counter machines. The union of the families of languages are denoted by  $\text{NCM} = \bigcup_{k, l \geq 0} \text{NCM}(k, l)$  and  $\text{DCM} = \bigcup_{k, l \geq 0} \text{DCM}(k, l)$ . Further,  $\text{DCA}$  is the family of languages accepted by one-way deterministic one counter machines (no reversal-bound). We will also sometimes refer to a multicounter machine as being in  $\text{NCM}(k, l)$  ( $\text{DCM}(k, l)$ ), if it has  $k$   $l$ -reversal bounded counters (and is deterministic). Note that a counter that makes  $l$  reversals can be simulated by  $\lceil \frac{l+1}{2} \rceil$  1-reversal-bounded counters. Hence, for example,  $\text{DCM}(1, 3) \subseteq \text{DCM}(2, 1)$ .

We denote by  $\text{REG}$  the family of regular languages, by  $\text{NPDA}$  the family of context-free languages, by  $\text{DPDA}$  the family of deterministic pushdown languages, by  $\text{DPDA}(l)$  the family of  $l$ -reversal-bounded

deterministic pushdown automata (with an upper bound of  $l$  on the number of changes between non-increasing and non-decreasing the size of the pushdown), by NPCM the family of languages accepted by nondeterministic pushdown automata augmented by a fixed number of reversal-bounded counters [4], and by DPCM the deterministic variant. We also denote by 2DCM the family of languages accepted by two-way input, deterministic finite automata (both a left and right input tape end-marker are required) augmented by reversal-bounded counters, and by 2DCM(1), 2DCM with one reversal-bounded counter [16]. A machine of this form is said to be *finite-crossing* if there is a fixed  $c$  such that the number of times the boundary between any two adjacent input cells is crossed is at most  $c$  [17]. A machine is *finite-turn* if the input head makes at most  $k$  turns on the input, for some  $k$ . Also, 2NCM is the family of languages accepted by two-way nondeterministic machines with a fixed number of reversal-bounded counters, while 2DPCM is the family of two-way deterministic pushdown machines augmented by a fixed number of reversal-bounded counters.

We give four examples below to illustrate the workings of the reversal-bounded counter machines.

**Example 1.** Let  $L = \{a^n b^n \mid n \geq 1\}$ . This language can be accepted by a DCM(1, 1) machine which reads  $a^n$  and stores  $n$  in the counter and then reads the  $b$ 's while decrementing the counter. It accepts if the counter becomes zero at the end of the input.

**Example 2.** Let  $L_k = \{x_1 \# \dots \# x_k \mid x_i \in \{a, b\}^+, x_j \neq x_k \text{ for } j \neq k\}$ . This can be accepted by a NCM( $k(k+1)/2, 1$ ), where we identify counter  $i$  by the name  $C_i$ , for  $1 \leq i \leq k(k+1)/2$ . The machine  $M_k$  operates by reading the input and verifying that for  $1 \leq i < j \leq k$ ,  $x_i$  and  $x_j$  are different in at least one position, or are different lengths, which is easy to check using the counters. To verify that they differ in at least one position, while scanning  $x_i$ ,  $M_k$  stores in counter  $C_i$  a nondeterministically guessed position  $p_i$  of  $x_i$  and records in the state the symbol  $a_{p_i}$  in that location. Then, when scanning  $x_j$ ,  $M_k$  stores in counter  $C_j$  a guessed location  $p_j$  of  $x_j$  and records in the state the symbol  $a_{p_j}$  in that location. At the end of the input, on stay transitions,  $M_k$  checks that  $a_{p_i} \neq a_{p_j}$  and  $p_i = p_j$  (by decrementing counters  $C_i$  and  $C_j$  simultaneously and verifying that they become zero at the same time).

**Example 3.** Let  $L = \{xx^R \mid x \in \{a, b\}^+, |x|_a > |x|_b\}$ , which is not a context-free language, can be accepted by an NPCM(2, 1)  $M$  (with counters labelled  $C_1$  and  $C_2$ ) which operates as follows: It scans the input and uses the pushdown to check that the input is of the form  $xx^R$ , by guessing the middle of the string, pushing  $x$ , and popping  $x$  while reading  $x^R$ . In parallel,  $M$  uses two counters  $C_1$  and  $C_2$  to store the numbers of  $a$ 's and  $b$ 's it encounters. Then, at the end of the input, on stay transitions,  $M$  decrements  $C_1$  and  $C_2$  simultaneously and verifies that the contents of  $C_1$  is larger than that of  $C_2$ . Clearly the counters are 1-reversal-bounded.

Note that the language  $L' = \{x \# x^R \mid x \in \{a, b\}^+, |x|_a > |x|_b\}$ , where  $\#$  is a new symbol, is in DPCM(2, 1).

**Example 4.**  $L = \{a^m b^n \mid m, n \geq 1, m \text{ is divisible by } n\}$  can be accepted by a 2DCM(1), which reads  $a^m$  and stores  $m$  in the counter. Then it checks that  $m$  is divisible by  $n$  by making left-to-right and right-to-left sweeps on  $b^n$  while decrementing the counter.

We note that each of NCM( $k, l$ ), NPCM( $k, l$ ), NCM, NPCM, NPDA, REG (for each  $k, l$ ) form full trios (discussed in Section 1) [4, 2], and are therefore immediately closed under left and right quotient with regular languages, prefix, suffix, infix, and outfix.

The next result proved in [14] gives examples of weak and strong machines that are equivalent over word-bounded languages.

**Theorem 3.** [14] The following are equivalent for every word-bounded language  $L$ :

1.  $L$  can be accepted by an NCM.
2.  $L$  can be accepted by an NPCM.
3.  $L$  can be accepted by a finite-crossing 2NCM.
4.  $L$  can be accepted by a DCM.

5.  $L$  can be accepted by a finite-turn 2DCM(1).
6.  $L$  can be accepted by a finite-crossing 2DPCM
7.  $L$  is bounded-semilinear.

We also need the following result in [16]:

**Theorem 4.** [16] *Let  $L \subseteq a^*$  be accepted by a 2NCM (not necessarily finite-crossing). Then  $L$  is regular, hence, semilinear.*

### 3. Closure of DCM Under Erasing Operations

First, we discuss the left quotient of DCM with finite sets.

**Proposition 5.** *DCM is closed under left quotient with finite languages.*

PROOF. It is clear that DCM is closed under left quotient with a single word. Then the result follows from closure of DCM under union [4].  $\square$

This is in contrast to DPDA, which is not even closed under left quotient with sets of multiple letters. Indeed, the language  $\{\#a^n b^n \mid n > 0\} \cup \{\$a^n b^{2n} \mid n > 0\}$  is a DPDA language, but taking the left quotient with  $\{\$, \#\}$  produces a language which is not a DPDA language [18].

Next, we show the closure of DCM under right quotient with languages accepted by any nondeterministic reversal-bounded multicounter machine, even when augmented with a pushdown store.

**Proposition 6.** *Let  $L_1 \in \text{DCM}$  and let  $L_2 \in \text{NPCM}$ . Then  $L_1 L_2^{-1} \in \text{DCM}$ .*

PROOF. Consider a DCM machine  $M_1 = (k_1, Q_1, \Sigma, \triangleleft, \delta_1, s_0, F_1)$  and NPCM machine  $M_2$  over  $\Sigma$  with  $k_2$  counters where  $L(M_1) = L_1$  and  $L(M_2) = L_2$ . A DCM machine  $M'$  will be constructed accepting  $L_1 L_2^{-1}$ .

Let  $\Gamma = \{a_1, \dots, a_{k_1}\}$  be new symbols. First, intermediate NPCM machines will be built, one for each state  $q$  of  $M_1$ . The intuition will be described first. The machine built for state  $q$  will accept all counter values (encoded over  $\Gamma$ ) of the form  $a_1^{p_1} \dots a_{k_1}^{p_{k_1}}$ , whereby there exists some  $x \in \Sigma^*$ , such that:

- $M_1$  accepts  $x$  starting from state  $q$  and counter values  $(p_1, \dots, p_{k_1})$ , and
- $M_2$  accepts  $x$ .

The key use of these machines is that they are all bounded languages, and therefore Theorem 3 applies, and they can all be effectively converted to a DCM machine. From these, our final deterministic machine  $M'$  can be built by simulating  $M_1$  until it hits the end-marker  $\triangleleft$  in some state  $q$ . It then deterministically simulates the *values that remain in the counters* with the intermediate DCM machine constructed for state  $q$ , accepting if this intermediate machine does. Indeed, it will therefore accept if and only if, from these counter values and state,  $M_1$  can continue to lead to acceptance on some word  $x$ , and  $x$  is also in  $L_2$ . Hence, the deterministic simulation of the intermediate machines is the key.

Formally, for each  $q \in Q_1$ , let  $M_c(q)$  be an intermediate  $k_1 + k_2$  counter (plus a pushdown) NPCM machine over  $\Gamma$  constructed as follows: on input  $a_1^{p_1} \dots a_{k_1}^{p_{k_1}}$ ,  $M_c(q)$  increments the first  $k_1$  counters to  $(p_1, \dots, p_{k_1})$ . Then  $M_c(q)$  nondeterministically guesses a word  $x \in \Sigma^*$  (by using only stay transitions of  $M_c(q)$ ) but simulating lettered transitions on each letter of  $x$  nondeterministically one at a time) and simulates  $M_1$  on  $x \triangleleft$  starting from state  $q$  and from the counter values of  $(p_1, \dots, p_{k_1})$  using the first  $k_1$  counters, while in parallel, simulating  $M_2$  on  $x$  using the next  $k_2$  counters and the pushdown. This is akin to the product automaton construction described in [4] showing NPCM is closed under intersection with NCM. Then  $M_c(q)$  accepts if both  $M_1$  and  $M_2$  accept.

**Claim 1.** *Let  $L_c(q) = \{a_1^{p_1} \dots a_{k_1}^{p_{k_1}} \mid \exists x \in L_2 \text{ such that } (q, x \triangleleft, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f, \triangleleft, p'_1, \dots, p'_{k_1}), p'_i \geq 0, 1 \leq i \leq k_1, q_f \in F_1\}$ . Then  $L(M_c(q)) = L_c(q)$ .*

PROOF. Consider  $w = a_1^{p_1} \dots a_{k_1}^{p_{k_1}} \in L_c(q)$ . Then there exists  $x$  where  $x \in L_2$  and  $(q, x \triangleleft, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f^1, \triangleleft, p'_1, \dots, p'_{k_1})$ , where  $q_f^1 \in F_1$ . There must then be some final state  $q_f^2 \in F_2$  reached when reading  $x \triangleleft$  in  $M_2$ . Then,  $M_c(q)$ , on input  $w$  places  $(p_1, \dots, p_{k_1}, 0, \dots, 0)$  on the counters and then can nondeterministically guess  $x$  letter-by-letter and simulate  $x$  in  $M_1$  from state  $q$  on the first  $k_1$  counters and simulate  $x$  in  $M_2$  from its initial configuration on the remaining counters and pushdown. Then  $M_c(q)$  ends up in state  $(q_f^1, q_f^2)$ , which is final. Hence,  $w \in L(M_c(q))$ .

Consider  $w = a_1^{p_1} \dots a_{k_1}^{p_{k_1}} \in L(M_c(q))$ . After adding each  $p_i$  to counter  $i$ ,  $M_c(q)$  guesses  $x$  and simulates  $M_1$  on the first  $k_1$  counters from  $q$  and simulates  $M_2$  on the remaining counters and the pushdown from an initial configuration. It follows that  $x \in L_2$ , and  $(q, x \triangleleft, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f^1, \triangleleft, p'_1, \dots, p'_{k_1})$ ,  $p'_i \geq 0$ ,  $1 \leq i \leq k_1$ ,  $q_f^1 \in F_1$ . Hence,  $w \in L_c(q)$ .  $\square$

Since for each  $q \in Q_1$ ,  $M_c(q)$  is in NPCM, it accepts a semilinear language [4], and since the accepted language is bounded, it is bounded-semilinear and can therefore be accepted by a DCM-machine by Theorem 3. Let  $M'_c(q)$  be this DCM machine, and let  $k'$  be the maximum number of counters of any DCM machine  $M'_c(q)$ ,  $q \in Q_1$ .

Thus, a final DCM machine  $M'$  with  $k_1 + k'$  counters is built as follows. In it,  $M'$  has  $k_1$  counters used to simulate  $M_1$ , and also  $k'$  additional counters, used to simulate some  $M'_c(q)$ , for some  $q \in Q_1$ . Then,  $M'$  reads its input  $x \triangleleft$ , where  $x \in \Sigma^*$ , while simulating  $M_1$  on the first  $k_1$  counters, either failing, or reaching some configuration  $(q, \triangleleft, p_1, \dots, p_{k_1})$ , for some  $q \in Q_1$ , upon first hitting the end-marker  $\triangleleft$ . If it does not fail, we then simulate the DCM-machine  $M'_c(q)$  on input  $a_1^{p_1} \dots a_{k_1}^{p_{k_1}}$ , but this simulation is done deterministically by subtracting 1 from the first  $k_1$  counters, in order, until each are zero instead of reading input characters, and accepts if  $a_1^{p_1} \dots a_{k_1}^{p_{k_1}} \in L(M'_c(q)) = L_c(q)$ . Then  $M'$  is deterministic, and accepts

$$\begin{aligned} & \{x \mid \text{either } (s_0, x \triangleleft, 0, \dots, 0) \vdash_{M_1}^* (q', a \triangleleft, p'_1, \dots, p'_{k_1}) \vdash_{M_1} (q, \triangleleft, p_1, \dots, p_{k_1}), \\ & \quad a \in \Sigma, \text{ or } (s_0, x \triangleleft, 0, \dots, 0) = (q, \triangleleft, p_1, \dots, p_{k_1}), \text{ s.t. } a_1^{p_1} \dots a_{k_1}^{p_{k_1}} \in L_c(q)\} \\ = & \{x \mid \text{either } (s_0, x \triangleleft, 0, \dots, 0) \vdash_{M_1}^* (q', a \triangleleft, p'_1, \dots, p'_{k_1}) \vdash_{M_1} (q, \triangleleft, p_1, \dots, p_{k_1}), \\ & \quad a \in \Sigma, \text{ or } (s_0, x \triangleleft, 0, \dots, 0) = (q, \triangleleft, p_1, \dots, p_{k_1}), \text{ where } \exists y \in L_2 \text{ s.t.} \\ & \quad (q, y \triangleleft, p_1, \dots, p_{k_1}) \vdash_{M_1}^* (q_f, \triangleleft, p''_1, \dots, p''_{k_1}), q_f \in F_1\} \\ = & \{x \mid xy \in L_1, y \in L_2\} \\ = & L_1 L_2^{-1}. \end{aligned}$$

$\square$

This immediately shows closure for the prefix operation.

**Corollary 7.** *If  $L \in \text{DCM}$ , then  $\text{pref}(L) \in \text{DCM}$ .*

We can modify this construction to show a strong closure result for one-counter languages that does not increase the number of counters.

**Proposition 8.** *Let  $l \in \mathbb{N}$ . If  $L_1 \in \text{DCM}(1, l)$  and  $L_2 \in \text{NPCM}$ , then  $L_1 L_2^{-1} \in \text{DCM}(1, l)$ .*

PROOF. The construction is similar to the one in Proposition 6. However, we note that since the input machine for  $L_1$  has only one counter,  $L_c(q)$  is unary (regardless of the number of counters needed for  $L_2$ ). Thus  $L_c(q)$  is unary and semilinear, and Parikh's theorem states that all semilinear languages are letter-equivalent to regular languages [19], and all unary semilinear languages are regular. Thus  $L_c(q)$  is regular, and can be accepted by a DFA.

We can then construct  $M'$  accepting  $L_1 L_2^{-1}$  as in Proposition 6 without requiring any additional counters or counter reversals, by transitioning to the DFA accepting  $L_c(q)$  when we reach the end of input at state  $q$ .  $\square$

**Corollary 9.** *Let  $l \in \mathbb{N}$ . If  $L \in \text{DCM}(1, l)$ , then  $\text{pref}(L) \in \text{DCM}(1, l)$ .*



In fact, the constructions of Propositions 6 and 8 can be generalized from NPCM to any class of automata that can be defined using Definition 10. A condition to describe these classes of automata is described in more detail in [20]. But we only define the condition below in a way specific to our use in this paper. Only the first two conditions are required for Corollary 11, while the third is required for Corollary 15.

**Definition 10.** A family of languages  $\mathcal{F}$  is said to be reversal-bounded counter augmentable if

- every language in  $\mathcal{F}$  is effectively semilinear,
- given DCM machine  $M_1$  with  $k$  counters, state set  $Q$  and final state set  $F$ , and  $L_2 \in \mathcal{F}$ , we can effectively construct, for each  $q \in Q$ , the following language in  $\mathcal{F}$ ,

$$\{a_1^{p_1} \cdots a_k^{p_k} \mid \exists x \in L_2 \text{ such that } (q, x \triangleleft, p_1, \dots, p_k) \vdash_{M_1}^* (q_f, \triangleleft, p'_1, \dots, p'_k), \\ p'_i \geq 0, 1 \leq i \leq k, q_f \in F\},$$

- given DCM machine  $M_1$  with  $k$  counters, state set  $Q$ , initial state  $q_0$ , and  $L_2 \in \mathcal{F}$ , we can effectively construct, for each  $q \in Q$ , the following language in  $\mathcal{F}$ ,

$$\{a_1^{p_1} \cdots a_k^{p_k} \mid \exists x \in L_2 \text{ such that } (q_0, x, 0, \dots, 0) \vdash_{M_1}^* (q, \lambda, p_1, \dots, p_k)\}.$$

**Corollary 11.** Let  $L_1 \in \text{DCM}$  and  $L_2 \in \mathcal{F}$ , a family of languages that is reversal-bounded counter augmentable. Then  $L_1 L_2^{-1} \in \text{DCM}$ . Furthermore, if  $L_1 \in \text{DCM}(1, l)$  for some  $l \in \mathbb{N}$ , then  $L_1 L_2^{-1} \in \text{DCM}(1, l)$ .

There are many reversal-bounded counter augmentable families that  $L_2$  could be from in this corollary, such as:

- MPCA's: one-way machines with  $k$  pushdowns where values may only be popped from the first non-empty stack, augmented by a fixed number of reversal-bounded counters [20].
- TCA's: nondeterministic Turing machines with a one-way read-only input and a two-way read-write tape, where the number of times the read-write head crosses any tape cell is finitely bounded, again augmented by a fixed number of reversal-bounded counters [20].
- QCA's: NFA's augmented with a queue, where the number of alternations between the non-deletion phase and the non-insertion phase is bounded by a constant [20], augmented by a fixed number of reversal-bounded counters.
- EPDA's: embedded pushdown automata, modelled around a stack of stacks, introduced in [21] augmented by a fixed number of reversal-bounded counters. These accept the languages of tree-adjoining grammars, a semilinear subset of the context-sensitive languages. As was stated in [20], we can augment this model with a fixed number of reversal-bounded counters and still get an effectively semilinear family.

Finally, the construction of Proposition 6 can be used to show that deterministic one counter languages (non-reversal-bounded) are closed under right quotient with NCM.

**Proposition 12.** Let  $L_1 \in \text{DCA}$ , and let  $L_2 \in \text{NCM}$ . Then  $L_1 L_2^{-1} \in \text{DCA}$ .

**PROOF.** Again, the construction is similar to Proposition 6. However, since the input machine for  $L_1$  has only one counter,  $L_c(q)$  is unary (regardless of the number of counters needed for  $L_2$ ). Then  $L_c(q)$  is unary and is indeed an NPCM language, as  $M_c(q)$  simulates  $M_1$ , this time using the unrestricted pushdown to simulate the potentially non-reversal-bounded counter of  $M_1$ , while simulating  $M_2$  on the reversal-bounded counters. Thus, because NPCM accept only semilinear languages [4],  $L_c(q)$  is in fact a regular language and can be accepted by a DFA.  $M'$  can then be constructed to accept  $L_1 L_2^{-1}$  without requiring any additional counters or counter reversals by transitioning to the DFA accepting  $L_c(q)$  when we reach the end of input at state  $q$ .  $\square$

Next, for the case of one-counter machines that make only one counter reversal, it will be shown that a DCM-machine that can accept their suffix and infix languages can always be constructed. However, in some cases, these resulting machines often require more than one counter. Thus, unlike prefix,  $\text{DCM}(1, 1)$  is not closed under suffix, left quotient, or infix. But, the result is in DCM.

As the proof is quite lengthy, we will give some intuition for the result first. First, DCM is closed under union [4] (following from closure under intersection and complement) and so the second statement of Proposition 13 follows from the first. For the first statement, an intermediate NPCM machine is constructed from  $L_1$  and  $L$  that accepts a language  $L^c$  (here,  $c$  is a superscript label rather than an exponent). This language contains words of the form  $qa^i$  where there exists some word  $w$  such that both  $w \in L_1$ , and also from the initial configuration of  $M$  (accepting  $L$ ), it can read  $w$  and reach state  $q$  with  $i$  on the counter. Then, it is shown that this language is actually a regular language, using the fact that all semilinear unary languages are regular. Then,  $\text{DCM}(1, 1)$  machines are created for every state  $q$  of  $M$ . These accept all words  $w$  such that  $qa^i \in L^c$ , and in  $M$ , from state  $q$  and counter  $i$  with  $w$  to read as input,  $M$  can reach a final state while emptying the counter. The fact that  $L^c$  is regular allows these machines to be created.

**Proposition 13.** *Let  $L \in \text{DCM}(1, 1)$ ,  $L_1 \in \text{NPCM}$ . Then  $L_1^{-1}L$  is the finite union of languages in  $\text{DCM}(1, 1)$ . Furthermore, it is in DCM.*

PROOF. For the first statement, let  $M_1$  be an NPCM machine accepting  $L_1$ , and let  $M = (1, Q, \Sigma, \triangleleft, \delta, q_0, F)$  be a 1-reversal bounded, 1-counter machine accepting  $L$ .

Next, we will argue that it is possible to assume, without loss of generality, that  $M$  has the following form:

1.  $Q = Q_\downarrow \cup Q_\uparrow$ ,
2. for all  $q \in Q_\downarrow$ , all transitions defined on  $q$  either decrease the counter or keep it the same,
3. for all  $q \in Q_\uparrow$ , all transitions defined on  $q$  either increase the counter or keep it the same,
4. the sequence of states  $p_0 p_1 \cdots p_n$ ,  $n \geq 0$ ,  $p_0 = q_0$ ,  $n \geq 0$  traversed in every computation from the initial configuration satisfies  $p_0 \cdots p_i \in Q_\uparrow^*$ ,  $p_{i+1} \cdots p_n \in Q_\downarrow^*$ ,  $0 \leq i \leq n$ , with the transition from  $p_{i+1}$  to  $p_{i+2}$  being the (first, if it exists) decreasing transition,
5. for all states  $q \in Q_\downarrow$  all stay transitions defined on  $q$  (except on  $\delta(q, \triangleleft, 0)$ ) change the counter,
6.  $\delta(q, d, 1)$  is defined for all  $q \in Q$ ,  $d \in \Sigma$ ,
7. the counter always empties before accepting.

Indeed, it is possible to transform a  $\text{DCM}(1, 1)$  machine of the form of  $M$  to another  $\text{DCM}(1, 1)$  machine  $\bar{M} = (1, \bar{Q}, \Sigma, \triangleleft, \bar{\delta}, q_0, \bar{F})$  satisfying the first four conditions, as follows. First, let  $Q_\uparrow = Q$ , and  $Q_\downarrow = \{q' \mid q \in Q\}$  (primed versions of the state set),  $\bar{Q} = Q_\uparrow \cup Q_\downarrow$ , and  $\bar{F} = F \cup \{q'_f \mid q_f \in F\}$ . Then, for all transitions that either decrease the counter or keep it the same,  $(p, T, j) \in \delta(q, c, i)$ ,  $i \in \{0, 1\}$ ,  $j \in \{0, -1\}$ , instead create the transition  $(p', T, j) \in \bar{\delta}(q', c, i)$ . Further, for all transitions of  $\delta$  that either increase the counter or keep it the same, keep this transition in  $\bar{\delta}$ . Then, the first three conditions are satisfied. Then, for all those decreasing transitions  $(p, T, -1) \in \delta(q, c, 1)$ , add in  $(q', S, 0) \in \bar{\delta}(q, c, 1)$ . Therefore, condition four is satisfied. It is clear that  $L(\bar{M}) = L(M)$ , as any sequence of transitions with at most one counter reversal (all accepting computations have at most one reversal) can traverse the same transitions (using states in  $Q_\uparrow$ ) until the first decrease transition, at which point only the new stay transitions from  $Q_\uparrow$  to  $Q_\downarrow$  are defined, and then the computation continues with transitions on  $Q_\downarrow$ . This machine can be further transformed into one accepting the same language and additionally satisfying conditions 5, 6, and 7 as any sequence of stay transitions that does not change the counter can be “skipped over” to either a right transition or a decrease transition, a “dead state” can be added to satisfy condition 6, and the states can enforce 7.

Therefore, assume without loss of generality that  $M$  satisfies these conditions. This will simplify the rest of the construction.



Next, we create an NPCM machine  $M'$  that accepts

$$L^c = \{qa^i \mid \exists w \in L_1, (q_0, w, 0) \vdash_M^* (q, \lambda, i)\},$$

where  $a$  is a new symbol not in  $\Sigma$ . Indeed,  $M'$  operates by nondeterministically guessing a word  $w \in \Sigma^*$ , letter-by-letter, and simulating in parallel (using stay transitions), the NPCM machine  $M_1$  using the pushdown and a set of counters, as well as simulating  $M$  on  $w$  on an additional counter. Then, after reading the last letter of the guessed  $w$ ,  $M'$  verifies that the simulated machine  $M$  is in state  $q$  (reading the state  $q$  as part of the input), and verifies that the contents of the simulated counter of  $M$  is  $i$ , matching the input. Then, it verifies that  $w$  is in  $L_1$  by continuing the simulation of  $M_1$  on the end-marker. Furthermore, for each  $q \in Q$ , the set  $q^{-1}L^c$  is a unary NPCM language (as discussed in Section 2, NPCM is closed under left quotient with regular languages). Indeed, every NPCM language is semilinear [4], and it is also known that every unary semilinear language is regular [19], and effectively constructable. Thus,  $L^c = \bigcup_{q \in Q} (q(q^{-1}L^c))$  is regular as well. Let  $M^c = (Q^c, Q \cup \{a\}, \delta^c, s_0^c, F^c)$  be a DFA accepting  $L^c$ . Assume without loss of generality that  $M^c$  is a complete DFA.

For the remainder of the proof, the layout will proceed by creating three sets of DCM(1, 1) machines and languages as follows:

1.  $M_0^q$ , for all  $q \in Q$ , and  $L_0^q = L(M_0^q)$ . We will construct it such that

$$L_0^q = \{w \mid (q, w \triangleleft, 0) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, qa^0 = q \in L^c\}. \quad (1)$$

2.  $M_\uparrow^q$ , for all  $q \in Q_\uparrow$ , and  $L_\uparrow^q = L(M_\uparrow^q)$ . We will construct it such that

$$L_\uparrow^q = \{w \mid \exists i > 0, (q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, qa^i \in L^c\}. \quad (2)$$

3.  $M_\downarrow^q$ , for all  $q \in Q_\downarrow$ , and  $L_\downarrow^q = L(M_\downarrow^q)$ . We will construct it such that

$$L_\downarrow^q = \{w \mid \exists i > 0, (q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, qa^i \in L^c\}. \quad (3)$$

It is clear that

$$L_1^{-1}L(M) = \bigcup_{q \in Q} L_0^q \cup \bigcup_{q \in Q_\uparrow} L_\uparrow^q \cup \bigcup_{q \in Q_\downarrow} L_\downarrow^q,$$

and thus it suffices to build the DCM(1, 1) machines and show that Equations (1), (2), and (3) hold. We will do this with each type next.

First, for (1), construct  $M_0^q$  for  $q \in Q$  as follows:  $M_0^q$  operates just like  $M$  starting at state  $q$  if  $q \in L^c$ , and if  $q \notin L^c$ , then it accepts  $\emptyset$ . Hence, (1) is true.

Next, we will show (3) is true. It will be shown that  $L_\downarrow^q$  is a regular language. Then the construction and proof of correctness of (3) will be used within the proof and construction of (2). A slight generalization of (3) will be used in order to accommodate its use for (2). Despite the languages being regular, NCM(1, 1) machines will be built to accept them, but without using the counter (for consistency of notation, and to use nondeterminism). It is immediate that these are regular, can be converted to NFAs, then to DFAs, then to DCM(1, 1) machines that do not use the counter. Intuitively, each NCM(1, 1) machine (for each  $q \in Q_\downarrow$ ) will simulate  $M$  starting at state  $q$ , but then only non-increasing transitions can be used, as only transitions on  $Q_\downarrow$  can be reached from  $q$ . However, instead of decreasing from a counter, the NCM(1, 1) machine instead simulates the DFA  $M^c$  in parallel, and reads a single  $a$  for every decrease of the simulated computation of  $M$ . If the simulated computation of  $M^c$  is in a final state, then the counter could be zero in the simulated computation of  $M$  and reach that configuration. But the simulated computation of  $M$  may only accept from configurations with larger counter values (depending on the remaining sequence of transitions). Thus, the new machine uses nondeterminism to try every possible configuration where zero could occur on the counter, trying each to see if the rest of the input accepts (by directly simulating  $M$ ).

We will give the construction here of the intermediate  $\text{NCM}(1, 1)$  machines that do not use the counter, then the proof of correctness of the construction. All the machines  $\overline{M_{\downarrow}^{q,q'}} \in \text{NCM}(1, 1)$ , for each  $q \in Q_{\downarrow}, q' \in Q$  will have the same set of input alphabets, states, transitions, and final states, with only the initial state differing.

Formally, let  $q \in Q_{\downarrow}, q' \in Q, q_0^c = \hat{\delta}^c(s_0^c, q')$ . Then  $\overline{M_{\downarrow}^{q,q'}} = (1, P_{\downarrow}, \Sigma, \triangleleft, \delta_{\downarrow}, s_{\downarrow}^{q,q'}, F_{\downarrow})$ , where  $P_{\downarrow} = (Q \times Q^c) \cup Q_{\downarrow}, s_{\downarrow}^{q,q'} = (q, q_0^c), F_{\downarrow} = F$ .

The transitions of  $\delta_{\downarrow}$  are created (none using the counter) by the following algorithm:

1. For all transitions  $(p, S, -1) \in \delta(r, d, 1), p, r \in Q_{\downarrow}, d \in \Sigma \cup \{\triangleleft\}$ , and all  $r^c \in Q^c$ , create

$$((p, \delta^c(r^c, a)), S, 0) \in \delta_{\downarrow}((r, r^c), d, 0),$$

and if  $\delta^c(r^c, a) \in F^c$ , create

$$(p, S, 0) \in \delta_{\downarrow}((r, r^c), d, 0).$$

2. For all transitions  $(p, R, 0) \in \delta(r, d, 1), p, r \in Q_{\downarrow}, d \in \Sigma$ , and all  $r^c \in Q^c$ , create

$$((p, r^c), R, 0) \in \delta_{\downarrow}((r, r^c), d, 0).$$

3. For all transitions  $(p, R, -1) \in \delta(r, d, 1), p, r \in Q_{\downarrow}, d \in \Sigma$ , and all  $r^c \in Q^c$ , create

$$((p, \delta^c(r^c, a)), R, 0) \in \delta_{\downarrow}((r, r^c), d, 0),$$

and if  $\delta^c(r^c, a) \in F^c$ , create

$$(p, R, 0) \in \delta_{\downarrow}((r, r^c), d, 0).$$

4. For all transitions  $(p, R, 0) \in \delta(r, d, 0), p, r \in Q_{\downarrow}, d \in \Sigma$ , create

$$(p, R, 0) \in \delta_{\downarrow}(r, d, 0).$$

5. For all transitions  $(p, S, 0) \in \delta(r, \triangleleft, 0), p, r \in Q_{\downarrow}$ , create

$$(p, S, 0) \in \delta_{\downarrow}(r, \triangleleft, 0).$$

The states of the machine consist of ordered pairs in  $Q \times Q^c$  in addition to states of  $Q_{\downarrow}$  to allow the simulation of a  $M$  and  $M^c$  in parallel, until the computation of  $M^c$  can hit a final state, at which point it can (optionally) switch to only simulating  $M$  on an empty counter. Transitions created in step 1 simulate all decreasing transitions that stay on the input, by reading an  $a$  from the simulation computation of  $M^c$  to simulate the decrease; and if the simulated computation of  $M^c$  can reach a final state, the simulated computation of  $M^c$  can optionally end. Transitions created in step 2 simulate all transitions of  $M$  defined on a positive counter that move right on the input but do not change the counter (thus not changing the state of  $Q^c$ ). Transitions created in step 3 simulate all transitions of  $M$  that move right on the input and decrease the counter by  $M^c$  reading  $a$  and optionally ending. Transitions created in step 4 and 5 simulate those of  $M$  defined on an empty counter verbatim.

Intuitively, the next claim demonstrates that it is possible to simulate  $M$  starting at  $q$  and counter value  $i$  whereby  $q'a^i \in L^c$  as follows: first it simulates the computation of  $M$  starting at  $q$  using the first component, and each decrease in counter reads  $a$  from the simulated computation of  $M^c$ . This continues until the counter reaches zero after  $i$  decreasing transitions, at which point the simulated computation of  $M^c$  is in a final state. Then, the simulation of  $M$  can continue verbatim. The formal proof is presented next.

**Claim 2.** For all  $q \in Q_{\downarrow}, q' \in Q$ ,

$$\{w \mid \exists i > 0, (q, w\triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, q'a^i \in L^c\} \subseteq L(\overline{M_{\downarrow}^{q,q'}}).$$

PROOF. Let  $q \in Q_\downarrow, q' \in Q$ . Let  $w$  be such that there exists  $i > 0, q_f \in F, q'a^i \in L^c$ , and  $(q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0)$ . Let  $p_j, w_j, x_j, 0 \leq j \leq m$  be such that  $p_0 = q, w = w_0, x_0 = i, q_f = p_m, w_m = \lambda, x_m = 0$  and  $(p_l, w_l \triangleleft, x_l) \vdash_M (p_{l+1}, w_{l+1} \triangleleft, x_{l+1}), 0 \leq l < m$ , via transition  $t_{l+1}$ . Then

$$(p_0, w_0 \triangleleft, x_0) \vdash_M^* (p_\gamma, w_\gamma \triangleleft, x_\gamma) \vdash_M^* (p_m, w_m \triangleleft, x_m),$$

where  $\gamma$  is the smallest number such that  $x_\gamma < i$  (it exists since  $i > 0$ ), and  $\mu$  the smallest number greater than or equal to  $\gamma$  such that  $x_\mu = 0$ .

The transitions  $t_1, \dots, t_{\gamma-1}$  are of the form, for  $0 \leq l < \gamma - 1$ ,  $(p_{l+1}, T_{l+1}, y_{l+1}) \in \delta(p_l, d_l, 1)$ , where  $i$  is on the counter on all  $x_0, \dots, x_{\gamma-1}$  (since  $x_0 = i$ , and  $x_\gamma$  is the first counter value less than  $\gamma$ ), and  $y_0, \dots, y_{\gamma-1}$  are all equal to 0. These must all be right transitions since they do not change the counter and so they create transitions in step 2 of the construction, of the form

$$((p_{l+1}, q_0^c), R, 0) \in \delta_\downarrow((p_l, q_0^c), d_l, 0),$$

for  $0 \leq l < \gamma - 1$ . Then,

$$((p_0, q_0^c), w_0 \triangleleft, x_0 - i = 0) \vdash_{M_\downarrow^{q, q'}}^* ((p_{\gamma-1}, q_0^c), w_{\gamma-1} \triangleleft, x_{\gamma-1} - i = 0).$$

The transitions  $t_\gamma, \dots, t_\mu$  are of the form, for  $\gamma - 1 \leq l < \mu$ ,  $(p_{l+1}, T_{l+1}, y_{l+1}) \in \delta(p_l, d_l, 1)$ , and for  $\gamma - 1 \leq l < \mu - 1$  ( $t_\mu$  is the last decreasing transition), creates transitions in steps 1, 2, and 3 of the form

$$((p_{l+1}, q_{l+1}^c), T_{l+1}, 0) \in \delta_\downarrow((p_l, q_l^c), d_l, 0),$$

for some  $q_l^c, q_{l+1}^c \in Q^c$ .

Then,

$$((p_{\gamma-1}, q_0^c), w_{\gamma-1} \triangleleft, 0) \vdash_{M_\downarrow^{q, q'}}^* \dots \vdash_{M_\downarrow^{q, q'}}^* ((p_{\mu-1}, q_{\mu-1}^c), w_{\mu-1} \triangleleft, 0),$$

where there are exactly  $i - 1$  decreasing transitions being simulated in this sequence. From  $q_{\mu-1}^c$ , reading one more  $a$ ,  $\delta^c(q_{\mu-1}^c, a) \in F^c$  since  $q'a^i \in F^c$ , and thus  $(p_\mu, T_\mu, y_\mu) \in \delta(p_{\mu-1}, d_{\mu-1}, 1)$  creates  $(p_\mu, T_\mu, y_\mu) \in \delta_\downarrow((p_{\mu-1}, q_{\mu-1}^c), d_{\mu-1}, 0)$  in step 1 or 3.

Then there remains transitions  $t_{\mu+1}, \dots, t_m$ , for  $\mu \leq l < m$  of the form  $(p_{l+1}, T_{l+1}, 0) \in \delta(p_l, d_l, 0)$ . These transitions are all in  $\delta_\downarrow$  and thus

$$(p_\mu, w_\mu \triangleleft, 0) \vdash_{M_\downarrow^{q, q'}}^* (p_m = q_f, \triangleleft, 0),$$

and hence  $w \in L(\overline{M_\downarrow^{q, q'}})$ .

□

The converse can be seen by examining an arbitrary computation of  $\overline{M_\downarrow^{q, q'}}$  which must have two components in the states, corresponding to a simulation of  $M$  in the first component and  $M^c$  in the second component, until some configuration where it switches to one component, continuing the simulation of  $M$ . The number of transitions used that simulate the reading of an  $a$  from the second component must be some  $i$ , where  $q'a^i \in L^c$ , and therefore a computation of  $M$  can proceed as in the simulation starting with  $i$  in the counter, and reach a final state. The formal proof is presented next.

**Claim 3.** For all  $q \in Q_\downarrow, q' \in Q$ ,

$$L(\overline{M_\downarrow^{q, q'}}) \subseteq \{w \mid \exists i > 0, (q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, q'a^i \in L^c\}.$$

PROOF. Let  $w \in L(\overline{M_{\downarrow}^{q,q'}})$ ,  $q \in Q_{\downarrow}$ ,  $q' \in Q$ . Let  $\mu$  ( $\mu$  is the last position of the derivation with an ordered pair as state),  $p_l, w_l, 0 \leq l \leq m$ , and  $q_j^c, 0 \leq j \leq \mu < m$  be such that  $p_0 = q, w_0 = w, w_m = \lambda, q_m \in F$ , and

$$((p_l, q_l^c), w_l \triangleleft, 0) \vdash_{\overline{M_{\downarrow}^{q,q'}}} ((p_{l+1}, q_{l+1}^c), w_{l+1} \triangleleft, 0),$$

for  $0 \leq l < \mu$ , via transition  $t_{l+1}$  of the form  $((p_{l+1}, q_{l+1}^c), T_{l+1}, 0) \in \delta_{\downarrow}((p_l, q_l^c), d_l, 0)$ , and

$$((p_{\mu}, q_{\mu}^c), w_{\mu} \triangleleft, 0) \vdash_{\overline{M_{\downarrow}^{q,q'}}} (p_{\mu+1}, w_{\mu+1} \triangleleft, 0),$$

via transition  $t_{\mu+1}$  of the form  $(p_{\mu+1}, T_{\mu+1}, 0) \in \delta_{\downarrow}((p_{\mu}, q_{\mu}^c), d_{\mu}, 0)$  and

$$(p_l, w_l \triangleleft, 0) \vdash_{\overline{M_{\downarrow}^{q,q'}}} (p_{l+1}, w_{l+1} \triangleleft, 0),$$

for  $\mu + 1 \leq l < m$  via transitions  $t_{l+1}$  of the form  $(p_{l+1}, T_{l+1}, 0) \in \delta_{\downarrow}(p_l, d_l, 0)$ . Let  $i$  be the number of times transitions created in step 1 or 3 are applied. Then by the transition  $t_{\mu+1}$ , this implies  $q'a^i \in F^c$ . Then, this implies that there are transitions  $(p_{l+1}, T_{l+1}, y_{l+1}) \in \delta(p_l, d_l, 1)$ , for all  $l, 0 \leq l \leq \mu$ , with  $i$  decreasing transitions and  $(p_{l+1}, T_{l+1}, 0) \in \delta(p_l, d_l, 0)$ , for all  $l, \mu + 1 \leq l < m$ , by the construction. Hence, the claim follows.  $\square$

We let  $M^{q,q'} = (1, Q^{q,q'}, \Sigma, \triangleleft, \delta_{\downarrow}^{q,q'}, s_{\downarrow}^{q,q'}, F_{\downarrow}^{q,q'})$  be a DCM(1,1) machine (that is hence deterministic) accepting  $L(\overline{M_{\downarrow}^{q,q'}})$  that never uses the counter, which can be created since it is regular. Assume all the sets of states of different machines  $Q_{\downarrow}^{q,q'}$  are disjoint.

Then, to prove Equation (3), only machines  $M_{\downarrow}^q = M_{\downarrow}^{q,q}, q \in Q_{\downarrow}$  accepting the languages  $L_{\downarrow}^{q,q}, q \in Q_{\downarrow}$  need to be considered, and they are all indeed regular.

The construction for  $M_{\uparrow}^q$  will be given next, and it will use the transitions from the machines  $M_{\downarrow}^{r,q}$  within it. Intuitively,  $M_{\uparrow}^q$  will simulate computations of  $M$  that start from configuration  $(q, u \triangleleft, i)$  up to a maximum counter value of  $\alpha$  and back to counter value  $i$  again. However, these computations are simulated by starting at a counter value of 0 instead of  $i$  (from  $(q, u \triangleleft, 0)$ ) to a maximum of  $\alpha - i$  (instead of  $\alpha$ ), back to 0 again (instead of  $i$ ), ending at a configuration of the form  $(r, u' \triangleleft, 0)$ . Thus, the simulated computation takes place with  $i$  subtracted from each counter value of each configuration. Then,  $M_{\uparrow}^q$  uses the machine  $M_{\downarrow}^{r,q}$  to test if the rest of the input can be accepted starting at  $r$  with any counter value that can reach  $q$  by using words in  $L^c$  that start with  $q$ .

Formally, for  $q \in Q_{\uparrow}$ ,  $M_{\uparrow}^q = (1, P_{\uparrow}, \Sigma, \triangleleft, \delta_{\uparrow}, s_{\uparrow}^q, F_{\uparrow})$ , where  $P_{\uparrow} = Q \cup \bigcup_{r \in Q} Q_{\downarrow}^{r,q}, s_{\uparrow}^q = q, F_{\uparrow} = \bigcup_{r \in Q_{\downarrow}} F^{r,q}$ , where  $Q$  is disjoint from other states.

The transitions of  $\delta_{\uparrow}$  are created by the following algorithm:

1. For all transitions  $(p, T, y) \in \delta(r, d, 1), p, r \in Q, d \in \Sigma \cup \{\triangleleft\}, T \in \{S, R\}, y \in \{-1, 0, 1\}$ , create

$$(p, T, y) \in \delta_{\uparrow}(r, d, e),$$

for both  $e = 1$ , and  $e = 0$  if  $r \in Q_{\uparrow}$ ,

2. Create  $(s_{\downarrow}^{r,q}, S, 0) \in \delta_{\uparrow}(r, d, 0)$ , for all  $d \in \Sigma \cup \{\triangleleft\}$ , and for all  $r \in Q_{\downarrow}$ ,

3. Add all transitions from  $M_{\downarrow}^{s,q}, s \in Q_{\downarrow}$ .

Indeed,  $M_{\uparrow}^q$  is deterministic as those transitions created in step 1 are in  $M$ , and  $M_{\downarrow}^{s,p}$  is deterministic, for all  $s, p$ .

**Claim 4.** For all  $q \in Q_{\uparrow}$ ,

$$\{w \mid \exists i > 0, (q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, qa^i \in L^c\} \subseteq L_{\uparrow}^q.$$

PROOF. Let  $q \in Q_\uparrow$ . Let  $w$  be such that there exists  $i > 0$ ,  $q_f \in F$ ,  $qa^i \in L^c$ , and  $(q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0)$ . Let  $p_j, w_j, x_j, 0 \leq j \leq m$  be such that  $p_0 = q, w = w_0, x_0 = i, q_f = p_m, \lambda = w_m, x_m = 0$  and  $(p_l, w_l \triangleleft, x_l) \vdash_M (p_{l+1}, w_{l+1} \triangleleft, x_{l+1}), 0 \leq l < m$ , via transition  $t_{l+1}$ . Assume that there exists  $\alpha > 1$  such that  $x_\alpha > i$ , and let  $\alpha$  be the smallest such number. Then, there exists

$$(p_0, w_0 \triangleleft, x_0) \vdash_M^* (p_\alpha, w_\alpha \triangleleft, x_\alpha) \vdash_M^* (p_\beta, w_\beta \triangleleft, x_\beta) \vdash_M^* (p_m, w_m \triangleleft, x_m),$$

where  $\beta$  is the smallest number bigger than  $\alpha$  such that  $x_\beta = i$ . In this case, in step 1 of the algorithm, transitions  $t_1, \dots, t_\alpha$  of the form  $(p_l, T_l, y_l) \in \delta(p_{l-1}, d_{l-1}, 1), 0 < l \leq \alpha$ , create transitions of the form  $(p_l, T_l, y_l) \in \delta_\uparrow(p_{l-1}, d_{l-1}, 0)$ , and thus

$$(p_0, w_0 \triangleleft, x_0 - i = 0) \vdash_{M_\uparrow^q}^* (p_{\alpha-1}, w_{\alpha-1} \triangleleft, x_{\alpha-1} - i = 0) \vdash_{M_\uparrow^q} (p_\alpha, w_\alpha \triangleleft, x_\alpha - i),$$

where  $x_\alpha - i > 0$ .

In step 1 of the algorithm, transitions  $t_{\alpha+1}, \dots, t_\beta$  of the form  $(p_l, T_l, y_l) \in \delta(p_{l-1}, d_{l-1}, 1), \alpha < l \leq \beta$  create transitions of the form

$$(p_l, T_l, y_l) \in \delta_\uparrow(p_{l-1}, d_{l-1}, 1).$$

Thus,  $(p_\alpha, w_\alpha \triangleleft, x_\alpha - i) \vdash_{M_\uparrow^q}^* (p_\beta, w_\beta \triangleleft, x_\beta - i = 0)$ , since  $x_\alpha - i, \dots, x_{\beta-1} - i$  are all greater than 0. Then, using a transition of type 2,  $(p_\beta, w_\beta \triangleleft, 0) \vdash_{M_\uparrow^q} (s_\downarrow^{p_\beta, q}, w_\beta \triangleleft, 0)$ . Then since  $(p_\beta, w_\beta \triangleleft, x_\beta) \vdash_M^* (p_m, \triangleleft, 0), p_m \in F$ , and  $p_\beta \in Q_\downarrow, qa^i \in L^c$ , then  $w_\beta \in L_\downarrow^{p_\beta, q}$ , by Claim 2. Hence,

$$(s_\downarrow^{p_\beta, q}, w_\beta \triangleleft, 0) \vdash_{M_\downarrow^{p_\beta, q}}^* (q'_f, \triangleleft, 0),$$

$q'_f \in F$ , and therefore, this occurs in  $M_\uparrow^q$  as well.

Lastly, the case where there does not exist an  $\alpha > i$  such that  $x_\alpha > i$  (thus  $i$  is the highest value in counter) is similar, by applying transitions of type 1 until the transitions before the first decrease (the first time a state from  $Q_\downarrow$  is reached), then a transitions of type 2, followed by a sequence of type 3 transitions as above.

□

The reverse containment can be shown by examining any accepting sequence of configurations, which has some initial simulation of  $M$ , followed by a computation of a machine  $M_\downarrow^{q', q}$ . The initial simulation can occur in  $M$  with  $i > 0$  added to each counter value, and the correctness of the remaining portion of  $M_\downarrow^{q', q}$  follows from Claim 3.

**Claim 5.** For all  $q \in Q_\uparrow$ ,

$$L_\uparrow^q \subseteq \{w \mid \exists i > 0, (q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, qa^i \in L^c\}.$$

PROOF. Let  $w \in L(M_\uparrow^q)$ . Then

$$(q, w \triangleleft, 0) \vdash_{M_\uparrow^q}^* (q', w' \triangleleft, 0) \vdash_{M_\uparrow^q} ((q', \delta^c(s_0^c, q)), w' \triangleleft, 0) \vdash_{M_\uparrow^q}^* (q'_f, \triangleleft, 0),$$

where  $q'_f \in F^{q', q}$ . Let  $\beta, p_l, w_l, x_l, 0 \leq l \leq \beta$  be such that  $p_0 = q, w_0 = w, x_0 = 0, q' = p_\beta, w' = w_\beta, x_\beta = 0$  such that  $(p_l, w_l \triangleleft, x_l) \vdash_{M_\uparrow^q} (p_{l+1}, w_{l+1} \triangleleft, x_{l+1}), 0 \leq l < \beta$ .

Then  $w' \in L_\downarrow^{q', q}$ , and therefore by Claim 3, there exists  $i > 0$  such that  $(q', w' \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0), q_f \in F, qa^i \in L^c$ . By the construction in step 1,

$$(p_0, w_0 \triangleleft, x_0 + i) \vdash_M \dots \vdash_M (p_\beta, w_\beta \triangleleft, x_\beta + i),$$

and since  $x_0 = x_\beta = 0$  and  $w' = w_\beta$  and  $q' = p_\beta$ , then  $(q, w \triangleleft, i) \vdash_M^* (q_f, \triangleleft, 0)$  and  $qa^i \in L^c$  and the claim follows.

□

Hence, Equation 2 holds.

It is also known that DCM is closed under union (by increasing the number of counters) [4]. Therefore, the finite union is in DCM. □

From this, we obtain the following general result.

**Proposition 14.** *Let  $L \in \text{DCM}(1, 1)$ ,  $L_1, L_2 \in \text{NPCM}$ . Then both  $(L_1^{-1}L)L_2^{-1}$  and  $L_1^{-1}(LL_2^{-1})$  are a finite union of languages in  $\text{DCM}(1, 1)$ . Furthermore, both languages are in DCM.*

PROOF. It will first be shown that  $(L_1^{-1}L)L_2^{-1}$  is the finite union of languages in  $\text{DCM}(1, 1)$ . Indeed,  $L_1^{-1}L$  is the finite union of languages in  $\text{DCM}(1, 1)$ ,  $1 \leq i \leq k$  by Proposition 13, and so  $L_1^{-1}L = \bigcup_{i=1}^k X_i$  for  $X_i \in \text{DCM}(1, 1)$ . Further, for each  $i$ ,  $X_i L_2^{-1}$  is the finite union of  $\text{DCM}(1, 1)$  languages by Proposition 8.

It remains to show that  $\bigcup_{i=1}^k X_i L_2^{-1} = (L_1^{-1}L)L_2^{-1}$ . If  $w \in \bigcup_{i=1}^k X_i L_2^{-1}$ , then  $w \in X_i L_2^{-1}$  for some  $i$ ,  $1 \leq i \leq k$ , then  $wy \in X_i$ ,  $y \in L_2$ . Then  $wy \in L_1^{-1}L$ , and  $w \in (L_1^{-1}L)L_2^{-1}$ . Conversely, if  $w \in (L_1^{-1}L)L_2^{-1}$ , then  $wy \in L_1^{-1}L$  for some  $y \in L_2$ , and so  $wy \in X_i$  for some  $i$ ,  $1 \leq i \leq k$ , and thus  $w \in X_i L_2^{-1}$ .

For  $L_1^{-1}(LL_2^{-1})$ , it is true that  $LL_2^{-1} \in \text{DCM}(1, 1)$  by Proposition 8. Then  $L_1^{-1}(LL_2^{-1})$  is the finite union of  $\text{DCM}(1, 1)$  by Proposition 13.

It is also known that DCM is closed under union (by increasing the number of counters) [4]. Therefore, both finite unions are in DCM. □

And, as with Corollary 11, this can be generalized to any language families that are reversal-bounded counter augmentable.

**Corollary 15.** *Let  $L \in \text{DCM}(1, 1)$ ,  $L_1 \in \mathcal{F}_1$ ,  $L_2 \in \mathcal{F}_2$ , where  $\mathcal{F}_1$  and  $\mathcal{F}_2$  are any families of languages that are reversal-bounded counter augmentable. Then  $(L_1^{-1}L)L_2^{-1}$  and  $L_1^{-1}(LL_2^{-1})$  are both a finite union of languages in  $\text{DCM}(1, 1)$ . Furthermore, both languages are in DCM.*

As a special case, when using the fixed regular language  $\Sigma^*$  for the right and left quotient, we obtain:

**Corollary 16.** *Let  $L \in \text{DCM}(1, 1)$ . Then  $\text{suff}(L)$  and  $\text{inf}(L)$  are both DCM languages.*

It is however sometimes necessary that the number of counters increase to accept  $\text{suff}(L)$  and  $\text{inf}(L)$ , when  $L \in \text{DCM}(1, 1)$  as seen from the next Proposition indicating that the suffix, infix, and outfix of a  $\text{DCM}(1, 1)$  language can be outside of  $\text{DCM}(1, 1)$ .

**Proposition 17.** *There exists  $L \in \text{DCM}(1, 1)$  where all of  $\text{suff}(L)$ ,  $\text{inf}(L)$ ,  $\text{outf}(L)$  are not in  $\text{DCM}(1, 1)$ .*

PROOF. Assume otherwise. Let  $L = \{a^n b^n c^n \mid n \geq 0\}$ ,  $L_1 = \{a^n b^n c^k \mid n, k \geq 0\}$ ,  $L_2 = \{a^n b^m c^m \mid n, m \geq 0\}$ ,  $L_3 = \{a^n b^m c^k \mid n, m, k \geq 0\}$ . Let  $\Sigma = \{a, b, c\}$  and  $\Gamma = \{d, e, f\}$ .

It is well-known that  $L$  is not a context-free language, and therefore is not a  $\text{DCM}(1, 1)$  language. However, each of  $L_1, L_2, L_3$  are  $\text{DCM}(1, 1)$  languages, and therefore, so are  $\overline{L_1}, \overline{L_2}, \overline{L_3}$  [4] and so is  $L' = d\#_1 \overline{L_1} \#_2 \cup e\#_1 \overline{L_2} \#_2 \cup f\#_1 \overline{L_3} \#_2$  (all complements with respect to  $\Sigma^*$ ). The symbols  $d, e, f$  are needed here, as each deterministically triggers the computation of a different  $\text{DCM}(1, 1)$  machine so that the resulting machine can be a  $\text{DCM}(1, 1)$  machine (although DCM is closed under union, this closure can increase the number of counters; but this type of marked union does not increase the number of counters). It can also be seen that  $\overline{L} = \overline{L_1} \cup \overline{L_2} \cup \overline{L_3}$ .

But  $\text{suff}(L') \cap \#_1 \Sigma^* \#_2 = \text{inf}(L') \cap \#_1 \Sigma^* \#_2 = \text{outf}(L') \cap \#_1 \Sigma^* \#_2 = \#_1 \overline{L} \#_2$ , and since  $\text{DCM}(1, 1)$  is closed under intersection with regular languages, under left and right quotient by a symbol, and under complement, this implies  $L$  is a  $\text{DCM}(1, 1)$  language, a contradiction. □



#### 4. Non-Closure Under Suffix, Infix, and Outfix for Multi-Counter and Multi-Reversal Machines

In [10], a technique was used to show that languages are not in  $\text{DCM} \cup 2\text{DCM}(1)$ . The technique uses undecidable properties to show non-closure. As  $2\text{DCM}(1)$  machines have a two-way input and a reversal-bounded counter, it is difficult to derive “pumping” lemmas for these languages. Furthermore, unlike  $\text{DCM}$  and  $\text{NCM}$  machines,  $2\text{DCM}(1)$  machines can accept non-semilinear languages. For example,  $L_1 = \{a^i b^k \mid i, k \geq 2, i \text{ divides } k\}$  can be accepted by a  $2\text{DCM}(1)$  whose counter makes only one reversal. However,  $L_2 = \{a^i b^j c^k \mid i, j, k \geq 2, k = ij\}$  cannot be accepted by a  $2\text{DCM}(1)$  [16]. This technique from [10] works as follows. The proof uses the fact that there is a recursively enumerable but not recursive language  $L_{\text{re}} \subseteq \mathbb{N}_0$  that is accepted by a deterministic 2-counter machine [22]. Here, these machines do not have an input tape, and acceptance is defined whereby  $n \in \mathbb{N}_0$  is accepted (i.e.,  $n \in L_{\text{re}}$ ) if and only if, when started with  $n$  in the first counter (encoded in unary) and zero in the second counter,  $M$  eventually halts (hence, acceptance is by halting).

Examining the constructions in [22] of the 2-counter machine demonstrates that the counters behave in a regular pattern. Initially one counter has some value  $d_1$  and the other counter is zero. Then, the machine’s operation can be divided into phases, where each phase starts with one of the counters equal to some positive integer  $d_i$  and the other counter equals 0. During the phase, the positive counter decreases, while the other counter increases. The phase ends with the first counter containing 0 and the other counter containing  $d_{i+1}$ . In the next phase, the modes of the counters are interchanged. Thus, a sequence of configurations where the phases are changing will be of the form:

$$(q_1, d_1, 0), (q_2, 0, d_2), (q_3, d_3, 0), (q_4, 0, d_4), (q_5, d_5, 0), (q_6, 0, d_6), \dots$$

where the  $q_i$ ’s are states, with  $q_1 = q_s$  (the initial state), and  $d_1, d_2, d_3, \dots$  are positive integers. The second component of the configuration refers to the value of the first counter, and the third component refers to the value of the second. Also, notice that in going from state  $q_i$  in phase  $i$  to state  $q_{i+1}$  in phase  $i+1$ , the 2-counter machine goes through intermediate states.

For each  $i$ , there are 5 cases for the value of  $d_{i+1}$  in terms of  $d_i$ :  $d_{i+1} = d_i, 2d_i, 3d_i, d_i/2, d_i/3$  (the division operation only occurs if the number is divisible by 2 or 3, respectively). The case applied is determined by  $q_i$ . Hence, a function  $h$  can be defined such that if  $q_i$  is the state at the start of phase  $i$ ,  $d_{i+1} = h(q_i)d_i$ , where  $h(q_i)$  is one of  $1, 2, 3, 1/2, 1/3$ .

Let  $T$  be a 2-counter machine accepting a recursively enumerable language that is not recursive. Assume that  $q_1 = q_s$  is the initial state, which is never re-entered, and if  $T$  halts, it does so in a unique state  $q_h$ . Let  $Q$  be the states of  $T$ , and 1 be a new symbol.

In what follows,  $\alpha$  is any sequence of the form  $\#I_1\#I_2\#\dots\#I_{2m}\#$  (thus we assume that the length is even), where for each  $i$ ,  $1 \leq i \leq 2m$ ,  $I_i = q1^k$  for some  $q \in Q$  and  $k \geq 1$ , represents a possible configuration of  $T$  at the beginning of phase  $i$ , where  $q$  is the state and  $k$  is the value of the first counter (resp., the second) if  $i$  is odd (resp., even).

Define  $L_0$  to be the set of all strings  $\alpha$  such that

1.  $\alpha = \#I_1\#I_2\#\dots\#I_{2m}\#$ ;
2.  $m \geq 1$ ;
3. for  $1 \leq j \leq 2m-1$ ,  $I_j \Rightarrow I_{j+1}$ , i.e., if  $T$  begins in configuration  $I_j$ , then after one phase,  $T$  is in configuration  $I_{j+1}$  (i.e.,  $I_{j+1}$  is a valid successor of  $I_j$ );

Then, the following was shown in [10].

**Lemma 18.**  $L_0$  is not in  $\text{DCM} \cup 2\text{DCM}(1)$ .

We will use this language exactly to show that taking either the suffix, infix, or outfit of a language in  $\text{DCM}(1, 3)$ ,  $\text{DCM}(2, 1)$ , or  $2\text{DCM}(1)$  can produce languages that are in neither  $\text{DCM}$  nor  $2\text{DCM}(1)$ .

**Proposition 19.** *There exists a language  $L \in \text{DCM}(1, 3)$  (respectively  $L \in \text{DCM}(2, 1)$ , and  $L \in 2\text{DCM}(1)$ ) such that  $\text{suff}(L) \notin \text{DCM} \cup 2\text{DCM}(1)$ ,  $\text{inf}(L) \notin \text{DCM} \cup 2\text{DCM}(1)$ , and  $\text{outf}(L) \notin \text{DCM} \cup 2\text{DCM}(1)$ .*

PROOF. Let  $L_0$  be the language defined above, which is not in  $\text{DCM} \cup 2\text{DCM}(1)$ . Let  $a, b$  be new symbols. Clearly,  $bL_0b$  is also not in  $\text{DCM} \cup 2\text{DCM}(1)$ . A configuration of  $T$  is any string of the form  $q1^k$ ,  $q \in Q, k \geq 1$  (whether it appears in any computation or not). Then let

$$L = \{a^i b \# I_1 \# I_2 \# \dots \# I_{2m} \# b \mid I_1, \dots, I_{2m} \text{ are configurations of 2-counter machine } T, i \leq 2m - 1, \\ I_{i+1} \text{ is not a valid successor of } I_i\}.$$

Clearly  $L$  is in  $\text{DCM}(1, 3)$ , in  $\text{DCM}(2, 1)$  (as  $\text{DCM}(1, 3)$  is a subset of  $\text{DCM}(2, 1)$  as mentioned in Section 2), and in  $2\text{DCM}(1)$  (as  $\text{DCM}(1, 3)$  is a subset of  $2\text{DCM}(1)$ ).

Let  $L_1$  be  $\text{suff}(L)$ . Suppose  $L_1$  is in  $\text{DCM}$  (resp.,  $2\text{DCM}(1)$ ). Then  $L_2 = \overline{L_1}$  is also in  $\text{DCM}$  (resp.,  $2\text{DCM}(1)$ ) since both are closed under complement [4, 16].

Let  $R = \{b \# I_1 \# I_2 \dots \# I_{2m} \# b \mid I_1, \dots, I_{2m} \text{ are configurations of } T\}$ . Then since  $R$  is regular,  $L_3 = L_2 \cap R$  is in  $\text{DCM}$  (resp.,  $2\text{DCM}(1)$ ) as both are closed under intersection with regular languages [4, 16]. We get a contradiction, since  $L_3 = bL_0b$ .

Non-closure under infix and outfix can be shown similarly (for outfix, the intersection with  $R$  enforces that only erasing of all of the  $a$ 's is considered).  $\square$

This implies non-closure under left-quotient with regular languages, and this result also extends to the embedding operation, a generalization of outfix.

**Corollary 20.** *There exists  $L \in \text{DCM}(1, 3)$  (respectively  $L \in \text{DCM}(2, 1)$ , and  $L \in 2\text{DCM}(1)$ ), and  $R \in \text{REG}$  such that  $R^{-1}L \notin \text{DCM} \cup 2\text{DCM}(1)$ .*

**Corollary 21.** *Let  $m > 0$ . Then there exists  $L \in \text{DCM}(1, 3)$  (respectively  $L \in \text{DCM}(2, 1)$ , and  $L \in 2\text{DCM}(1)$ ) such that  $\text{emb}(L, m) \notin \text{DCM} \cup 2\text{DCM}(1)$ .*

The results of Proposition 19 and Corollary 20 are optimal for suffix and infix as these operations applied to  $\text{DCM}(1, 1)$  are always in  $\text{DCM}$  by Corollary 16 (and since  $\text{DCM}(1, 2) = \text{DCM}(1, 1)$ ). But whether the outfix and embedding operations applied to  $\text{DCM}(1, 1)$  languages is always in  $\text{DCM}$  is an open question.

## 5. Closure and Non-Closure for NPCM, DPCM, and DPDA

To start, we consider quotients of nondeterministic classes, then use these results for contrast with deterministic classes.

**Proposition 22.** *Let  $\mathcal{L}_1$  and  $\mathcal{L}_2$  be classes of languages where  $\mathcal{L}_1$  is a full trio closed under intersection with languages in  $\mathcal{L}_2$ , and where  $L \in \mathcal{L}_2$  implies  $\Sigma^* \# L, L \# \Sigma^* \in \mathcal{L}_2$ , for an alphabet  $\Sigma$  and new symbol  $\#$ . Then  $\mathcal{L}_1$  is closed under left and right quotient with  $\mathcal{L}_2$ .*

PROOF. For right quotient, let  $L_1 \in \mathcal{L}_1, L_2 \in \mathcal{L}_2$ . If  $L_1 \in \mathcal{L}_1$ , then using an inverse homomorphism (where the homomorphism is from  $(\Sigma \cup \{\#\})^*$  to  $\Sigma^*$  that erases  $\#$  and fixes all other letters), and intersection with the regular language  $\Sigma^* \# \Sigma^*$ , it follows that  $L'_1 = \{x \# y \mid xy \in L_1\}$  is also in  $\mathcal{L}_1$ . Let  $L'_2 = \Sigma^* \# L_2 \in \mathcal{L}_2$ . Then  $L = L'_1 \cap L'_2 \in \mathcal{L}_1$ . Then, as every full trio is closed under gsm mappings, it follows that  $L_1 L_2^{-1} \in \mathcal{L}_1$  by erasing everything starting at the  $\#$  symbol.

Similarly with left quotient.  $\square$

**Corollary 23.** *NPCM (NCM respectively) is closed under left and right quotient with NCM.*

This follows since NPCM is a full trio closed under intersection with NCM [4], and NCM is closed under concatenation.

The question remains as to whether this is also true for deterministic machines instead. For machines with a stack, we have:

**Proposition 24.** *The right quotient of a DPDA(1) language (i.e., deterministic linear context-free) with a DCM(2, 1) language is not necessarily an NPDA language.*

PROOF. Take the DPDA(1) language  $L_1 = \{d^l c^k b^j a^i \# a^i b^j c^k d^l \mid i, j, k, l > 0\}$ . Take the DCM(2, 1) language  $L_2 = \{a^i b^j c^i d^j \mid i, j > 0\}$ . This is clearly a non-context-free language that is in DCM(2, 1). However,  $L_1 L_2^{-1} = L_2^R$ , which is also not context-free.  $\square$

Next we see that, in contrast to DCM and DPDA, DPCM is closed under neither prefix nor suffix. Indeed, both DCM and DPDA are closed under prefix (and right quotient with regular sets), but not left quotient with regular sets. Yet combining their stores into one type of machine yields languages that are closed under neither.

**Proposition 25.** *DPCM is not closed under prefix or suffix.*

PROOF. Assume otherwise. Let  $L$  be a language in NCM(1, 1) that is not in DPCM, which was shown to exist [23]. Let  $M$  be an NCM(1, 1) machine accepting  $L$ . Let  $T$  be a set of labels associated bijectively with transitions of  $M$ . Consider the language  $L' = \{t_m \cdots t_1 \$ w \mid M \text{ accepts } w \text{ via transitions } t_1, \dots, t_m \in T\}$ . This language is in DPCM since a DPCM machine  $M'$  can be built that first pushes  $t_m \cdots t_1$ , and then simulates  $M$  deterministically on transitions  $t_1, \dots, t_m$  while popping from the pushdown and reading  $w$ . Then  $\text{suff}(L') \cap \$\Sigma^* = \$L$ , a contradiction, as DPCM is clearly closed under left quotient with a single symbol.

Similarly for prefix, consider  $L^R$ , and create a machine  $M^R$  accepting  $L^R$ , which is possible since NCM(1, 1) is closed under reversal. Then  $L'' = \{w \$ t_1 \cdots t_m \mid M^R \text{ accepts } w^R \text{ via } t_1, \dots, t_m \in T\}$ . This is also a DPCM language as one can construct a machine  $M''$  that pushes  $w$ , then while popping  $w^R$  letter-by-letter, simulates  $M$  deterministically on transitions  $t_1, \dots, t_m$  on  $w^R$ . Then  $\text{pref}(L'') \cap \Sigma^* \$ = L \$$ , a contradiction, as DPCM is clearly closed under right quotient with a single symbol.  $\square$

**Corollary 26.** *DPCM is not closed under right or left quotient with regular sets.*

Thus, the deterministic variant of Corollary 23 gives non-closure.

The following is also evident from the proof of Proposition 25.

**Corollary 27.** *Every NCM language can be obtained by taking the right quotient (resp. left quotient) of a DPCM language by a regular language.*

The statement of this corollary cannot be weakened to taking the quotients of a DPDA with a regular language, since DPDA is closed under right quotient with regular languages [19].

Lastly, we will address the question of whether the left or right quotient of a DPDA language with a DCM language is always in DPCM.

**Proposition 28.** *The right quotient (resp. left quotient) of a DPDA(1) language with a DCM(1, 1) language can be outside DPCM.*

PROOF. To start, it is known that there exists an NCM(1, 1) language that is not in DPCM [23]. Let  $L$  be such a language, and let  $M$  be a NCM(1, 1) machine accepting  $L$ . Then  $L^R$  is also an NCM(1, 1) language, and let  $M^R$  be an NCM(1, 1) machine accepting it. Let  $T$  be a set of labels associated bijectively with transitions of  $M^R$ .

Then, we can create a DCM(1, 1) machine  $M'$  accepting words in  $\#(\Sigma \cup T)^*$  such that after reading  $\#$ ,  $M'$  simulates  $M^R$  deterministically by reading a label  $t \in T$  before simulating  $t$  deterministically. That is, if  $M'$  reads a letter  $a \in \Sigma$ ,  $M'$  stores it in a buffer (that can hold exactly one letter), and if  $M'$  reads a letter  $t \in T$ ,  $M'$  simulates  $M^R$  on the letter  $a$  in the buffer using transition  $t$ , completely deterministically. Then if  $t$  is a stay transition, the next letter must be in  $T$ , and the buffer stays intact, whereas if  $t$  is a right transition, then the buffer is cleared, and the next letter must be in  $\Sigma$ . If the input is not of this form (for example, if there are two letters from  $\Sigma$  in a row, or a transition label representing a right move followed by

another transition label), the machine crashes (cannot continue and does not accept). The first transition must also be from the initial state, and the simulation must end in a final state. It is clear then that if  $h$  is a homomorphism that erases letters of  $T$  and fixes letters of  $\Sigma$ , then  $h(L(M')) = L(M^R)$ .

Then, consider the language  $L_1 = \{w\#x \mid w \in \Sigma^*, x \in (\Sigma \cup T)^*, h(x) = w^R\}$ . Then  $L_1 \in \text{DPDA}(1)$ .

Consider  $L_2 = L_1 L(M')^{-1}$ . Then  $L_2 = \{w \mid w \in \Sigma^*, \text{ there exists } x \in (\Sigma \cup T)^* \text{ such that } h(x) = w^R, \text{ and } h(x) \in L(M^R)\}$ . Hence,  $L_2 = \{w \mid w \in \Sigma^*, w \in L(M)\} = L$ , which is not in DPCM.

Similarly for left quotient by using the DPDA(1) language  $L_1 = \{x\#w \mid w \in \Sigma^*, x \in (\Sigma \cup T)^*\}$ .  $\square$

The following is also evident from the proof above.

**Corollary 29.** *Every NCM language can be obtained by taking the right quotient (resp. left quotient) of a DPDA(1) language by a DCM language.*

Again, this statement cannot be weakened to the right quotient of a DPDA with a regular language since DPDA languages are closed under right quotient with regular languages [18].

## 6. Right and Left Quotients of Regular Sets

Let  $\mathcal{F}$  be any family of languages (which need not be recursively enumerable). It is known that REG is closed under right quotient by languages in  $\mathcal{F}$  [13]. However, this closure need not be effective, as it will depend on the properties of  $\mathcal{F}$ . The following is an interesting observation which connects decidability of the emptiness problem to effectiveness of closure under right quotient:

**Proposition 30.** *Let  $\mathcal{F}$  be any family of languages which is effectively closed under intersection with regular sets and whose emptiness problem is decidable. Then REG is effectively closed under both left and right quotient by languages in  $\mathcal{F}$ .*

PROOF. We will start with right quotient.

Let  $L_1 \in \text{REG}$  and  $L_2$  be in  $\mathcal{F}$ . Let  $M$  be a DFA accepting  $L_1$ . Let  $q$  be a state of  $M$ , and  $L_q = \{y \mid M \text{ from initial state } q \text{ accepts } y\}$ . Let  $Q' = \{q \mid q \text{ is a state of } M, L_q \cap L_2 \neq \emptyset\}$ . Since  $\mathcal{F}$  is effectively closed under intersection with regular sets and has a decidable emptiness problem,  $Q'$  is computable. Then a DFA  $M'$  accepting  $L_1 L_2^{-1}$  can be obtained by just making  $Q'$  the set of accepting states in  $M$ .

Next, for left quotient, let  $L_1$  be in  $\mathcal{F}$ , and  $L_2$  in REG be accepted by a DFA  $M$  whose initial state is  $q_0$ .

Let  $L_q = \{x \mid M \text{ on input } x \text{ ends in state } q\}$ . Let  $Q' = \{q \mid q \text{ is a state of } M, L_q \cap L_1 \neq \emptyset\}$ . Then  $Q'$  is computable, since  $\mathcal{F}$  is effectively closed under intersection with regular sets and has a decidable emptiness problem.

We then construct an NFA (with  $\lambda$ -transitions)  $M'$  to accept  $L_1^{-1} L_2$  as follows:  $M'$  starting in state  $q_0$  with input  $y$ , then nondeterministically goes to a state  $q$  in  $Q'$  without reading any input, and then simulates the DFA  $M$ .

$\square$

**Corollary 31.** *REG is effectively closed under left and right quotient by languages in:*

1. *the families of languages accepted by NPCM and 2DCM(1) machines,*
2. *the family of languages accepted by MPCAs, TCAs, QCAs, and EPDAs,*
3. *the families of ETOL and Indexed languages.*

PROOF. These families are closed under intersection with regular sets. They have also a decidable emptiness problem [20, 24, 25]. The family of ETOL languages and Indexed languages are discussed further in [25] and [24] respectively.  $\square$

## 7. Closure for Bounded Languages

In this subsection, deletion operations applied to bounded and letter-bounded languages will be examined. We will need the following corollary to Theorem 4.

**Corollary 32.** *Let  $L \subseteq \#a^*\#$  be accepted by a 2NCM. Then  $L$  is regular.*

**Proposition 33.** *If  $L$  is a bounded language accepted by either a finite-crossing 2NCM, an NPCM or a finite-crossing 2DPCM, then all of  $\text{pref}(L)$ ,  $\text{suff}(L)$ ,  $\text{inf}(L)$ ,  $\text{outf}(L)$  can be accepted by a DCM.*

PROOF. By Theorem 3, an NCM can be constructed that accepts  $L$ . Further, one can construct NCM's accepting  $\text{pref}(L)$ ,  $\text{suff}(L)$ ,  $\text{inf}(L)$ ,  $\text{outf}(L)$  since one-way NCM is closed under prefix, suffix, infix and outfix. In addition, it is known that applying these operations on bounded languages produce only bounded languages. Thus, by another application of Theorem 3, the result can then be converted to a DCM.  $\square$

The “finite-crossing” requirement in the proposition above is necessary:

**Proposition 34.** *There exists a letter-bounded language  $L$  accepted by a 2DCM(1) machine which makes only one reversal on the counter such that  $\text{suff}(L)$  (resp.,  $\text{inf}(L)$ ,  $\text{outf}(L)$ ,  $\text{pref}(L)$ ) is not in  $\text{DCM} \cup 2\text{DCM}(1)$ .*

PROOF. Let  $L = \{a^i\#b^j\# \mid i, j \geq 2, j \text{ is divisible by } i\}$ . Clearly,  $L$  can be accepted by a 2DCM(1) which makes only one reversal on the counter. If  $\text{suff}(L)$  is in  $\text{DCM} \cup 2\text{DCM}(1)$ , then  $L' = \text{suff}(L) \cap \#b^+\#$  would be in  $\text{DCM} \cup 2\text{DCM}(1)$ . From Corollary 32, we get a contradiction, since  $L'$  is not semilinear. The other cases are shown similarly.  $\square$

## 8. Conclusions

We investigated many different deletion operations applied to languages accepted by one-way and two-way deterministic reversal-bounded multicounter machines, deterministic pushdown automata, and finite automata. The operations include the prefix, suffix, infix, and outfix operations, as well as left and right quotient with languages from different families. Although it is frequently expected that language families defined from deterministic machines will not be closed under deletion operations, we showed that DCM is closed under right quotient with languages from many different language families, such as the context-free languages. When starting with one-way deterministic machines with one counter that makes only one reversal ( $\text{DCM}(1, 1)$ ), taking the left quotient with languages from many different language families, such as the context-free languages, yields only languages in DCM (by increasing the number of counters). It follows from these results that the suffix or infix closure of  $\text{DCM}(1, 1)$  languages are all in DCM. These results are surprising given the nondeterministic behaviour of the deletion operations. However, for both  $\text{DCM}(1, 3)$ , or  $\text{DCM}(2, 1)$ , taking the left quotient (or even just the suffix operation) yields languages that can neither be accepted by deterministic reversal-bounded multicounter machines, nor by 2-way nondeterministic machines with one reversal-bounded counter ( $2\text{DCM}(1)$ ).

Some interesting open questions remain. For example, is the outfix and embedding operations applied to  $\text{DCM}(1, 1)$  languages always yield languages in DCM? Also, other deletion operations, such as schema for parallel deletion [12] have not yet been investigated applied to languages accepted by deterministic machines.

## Acknowledgements

We thank the referees for helpful suggestions improving the presentation of the paper.

## References

- [1] J. Eremondi, O. H. Ibarra, I. McQuillan, Deletion operations on deterministic families of automata, in: R. Jain, S. Jain, F. Stephan (Eds.), *Lecture Notes in Computer Science*, Vol. 9076 of 12th Annual Conference on Theory and Applications of Models of Computation, TAMC 2015, Singapore, 2015, pp. 388–399.
- [2] S. Ginsburg, *Algebraic and Automata-Theoretic Properties of Formal Languages*, North-Holland Publishing Company, Amsterdam, 1975.
- [3] B. S. Baker, R. V. Book, Reversal-bounded multipushdown machines, *Journal of Computer and System Sciences* 8 (3) (1974) 315–332.
- [4] O. H. Ibarra, Reversal-bounded multicounter machines and their decision problems, *Journal of the ACM* 25 (1) (1978) 116–133.
- [5] O. H. Ibarra, On strong reversibility in P Systems and related problems, *International Journal of Foundations of Computer Science* 22 (01) (2011) 7–14.
- [6] O. H. Ibarra, J. Su, Z. Dang, T. Bultan, R. A. Kemmerer, Counter machines and verification problems, *Theoretical Computer Science* 289 (1) (2002) 165–189.
- [7] R. Alur, J. V. Deshmukh, Nondeterministic streaming string transducers, in: L. Aceto, M. Henzinger, J. Sgall (Eds.), *Automata, Languages and Programming*, Vol. 6756 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 1–20.
- [8] M. Hague, A. W. Lin, Model checking recursive programs with numeric data types, in: G. Gopalakrishnan, S. Qadeer (Eds.), *Computer Aided Verification*, Vol. 6806 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2011, pp. 743–759.
- [9] G. Xie, Z. Dang, O. H. Ibarra, A solvable class of quadratic diophantine equations with applications to verification of infinite-state systems, in: J. C. Baeten, J. K. Lenstra, J. Parrow, G. J. Woeginger (Eds.), *Automata, Languages and Programming*, Vol. 2719 of *Lecture Notes in Computer Science*, Springer Berlin Heidelberg, 2003, pp. 668–680.
- [10] J. Eremondi, O. Ibarra, I. McQuillan, Insertion operations on deterministic reversal-bounded counter machines, in: A. Dediu, E. Formenti, C. Martín-Vide, B. Truthe (Eds.), *Lecture Notes in Computer Science*, Vol. 8977 of 9th International Conference on Language and Automata Theory and Applications, LATA 2015, Nice, France, 2015, pp. 200–211.
- [11] E. Chiniforooshan, M. Daley, O. H. Ibarra, L. Kari, S. Seki, One-reversal counter machines and multihead automata: Revisited, *Theoretical Computer Science* 454 (2012) 81–87.
- [12] L. Kari, S. Seki, Schema for parallel insertion and deletion: Revisited, *International Journal of Foundations of Computer Science* 22 (07) (2011) 1655–1668.
- [13] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, Reading, MA, 1979.
- [14] O. H. Ibarra, S. Seki, Characterizations of bounded semilinear languages by one-way and two-way deterministic machines, *International Journal of Foundations of Computer Science* 23 (6) (2012) 1291–1306.
- [15] H. Jürgensen, L. Kari, G. Thierrin, Morphisms preserving densities, *International Journal of Computer Mathematics* 78 (2001) 165–189.
- [16] O. H. Ibarra, T. Jiang, N. Tran, H. Wang, New decidability results concerning two-way counter machines, *SIAM J. Comput.* 23 (1) (1995) 123–137.
- [17] E. M. Gurari, O. H. Ibarra, The complexity of decision problems for finite-turn multicounter machines, *Journal of Computer and System Sciences* 22 (2) (1981) 220–229.
- [18] S. Ginsburg, S. Greibach, Deterministic context free languages, *Information and Control* 9 (6) (1966) 620–648.
- [19] M. Harrison, *Introduction to Formal Language Theory*, Addison-Wesley series in computer science, Addison-Wesley Pub. Co., 1978.
- [20] T. Harju, O. Ibarra, J. Karhumäki, A. Salomaa, Some decision problems concerning semilinearity and commutation, *Journal of Computer and System Sciences* 65 (2) (2002) 278–294.
- [21] K. Vijayashanker, A study of tree adjoining grammars, Ph.D. thesis, Philadelphia, PA, USA (1987).
- [22] M. L. Minsky, Recursive unsolvability of Post’s problem of “tag” and other topics in theory of Turing Machines, *Annals of Mathematics* 74 (3) (1961) pp. 437–455.
- [23] O. H. Ibarra, Visibly pushdown automata and transducers with counters, *Fundamenta Informaticae*. To appear. (2016).
- [24] A. V. Aho, Indexed grammars—an extension of context-free grammars, *J. ACM* 15 (4) (1968) 647–671.
- [25] G. Rozenberg, A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, Inc., New York, 1980.